빅콘테스트 2017 챌린지 리그

– 빅데이터 분석을 통한 대출상환 예측 알고리즘 개발 –

Ensemble Method Using Greedy DBSCAN Up&Dowan Sampling

투빅스

고려대학교 산업경영공학 대학원 서덕성

연세대학교 산업공학 대학원 이경택

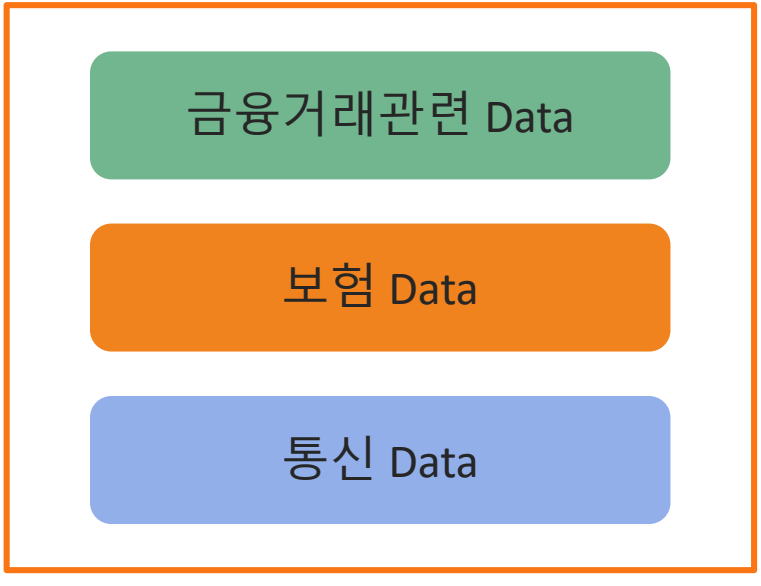
연세대학교 산업공학 대학원 김상진

INDEX

01 Data Preprocessing

주제 정의 및 데이터 소개

- 대출 연체의 여부를 고객 관련 금융, 통신, 보험 데이터를 통해 효과적으로 예측하는 신용평가 모델 알고리즘을 개발



분석에 앞서 분석의 성능을 높이기 위한 파생변수 생성 및 데이터 정제 실시

01 Data Preprocessing

One-hot encoding을 활용한 범주형 변수 정제

- 범주형 변수의 경우에는 아래의 예제처럼 one-hot encoding 방법으로 바꾸었다.

  ex) 왼쪽의 표에서 LINE_STUS 변수에 대한 값을 one-hot encoding 방법으로 바꿀 시 오른쪽 표와 같다.

CUST_ID	LINE_STUS (회신 상태)
1	S(정지)
2	S(정지)
3	U(사용)
⋮	⋮

CUST_ID	…	회신 상태 S	회신 상태 U	…
1	…	1	0	…
2	…	1	0	…
3	…	0	1	…
⋮	⋮	⋮	⋮	⋮

01 Data Preprocessing

Gradient Boosting model을 기반으로 한 파생변수 추출

1) 전체 data를 가지고 gradient boosting model을 사용하여 주요변수 추출
2) 상위 30개의 변수를 조합하여 새로운 파생변수 생성
3) 변수 Table을 통해 유의미하게 보여지는 파생변수를 추출
4) gradient boosting model를 다시 사용하여 파생변수가 중요 변수로 채택 시 최종 채택 (총 82개 생성)

주요 파생변수 의미	변수 생성 공식
은행권 대출 비율	$\frac{\text{산출일 기준 총 은행권 금액}}{\text{산출일 기준 총 대출 금액}}$
소득대비 대출 비율	$\frac{\text{산출일 기준 총 신용대출 금액}}{\text{직업정보기반 추정 소득금액}}$
소득대비 연체 비율	$\frac{\text{경과월수 중 연체경험월수의 비율}}{\text{직업정보기반 추정 소득금액}}$
신용등급 대비 보험금 지급 비율	$\frac{\text{가계합산 보험금지급 총액}}{\text{한화생명에서 실행된 가장 최근 대출시점의 신용 등급}}$
총 납입 보험료 대비 보험금 지급 비율	$\frac{\text{가계합산 보험금지급 총액}}{\text{가계합산 기준 유효한 계약의 총 납입 보험료}}$
⋮	⋮

01 Data Preprocessing

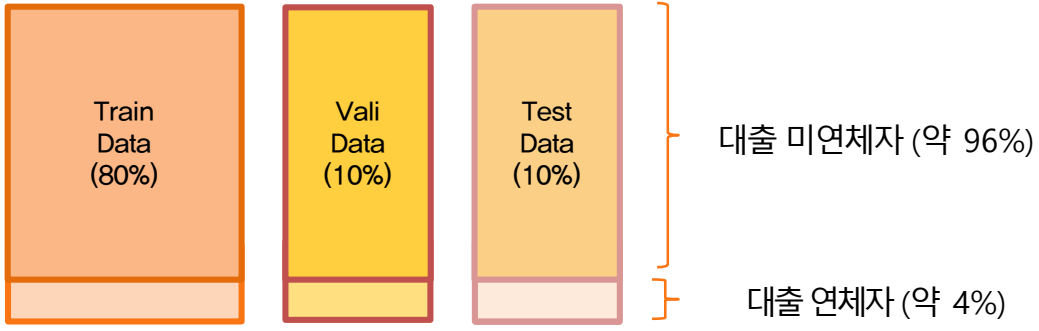
결측값 처리

- 추출된 파생변수 중 NaN, Inf 값들은 분모에 0이 들어가거나 분자 분모에 동시에 0이 들어가는 경우에 발생하는 문제이므로 이를 0으로 전처리하여 최종적인 Dataset을 완성

CUST_ ID	기존 변수 + 파생변수+Y
1	
2	
⋮	100,233 X 215 Data Frame
100,232	
100,233	

02 Modeling

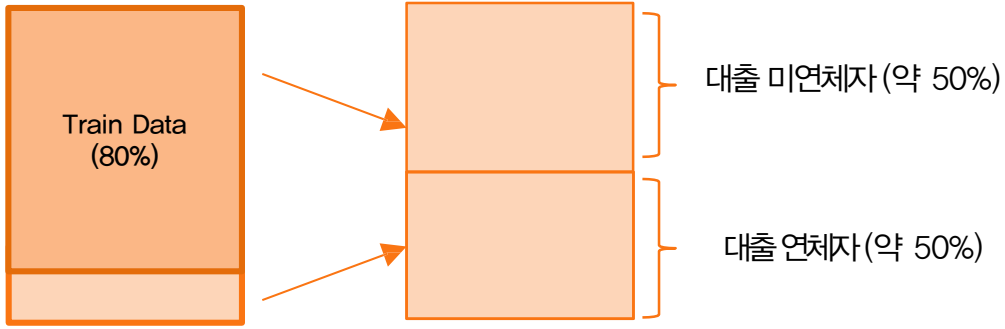
Data split



전체 데이터에서 대출 미연체자와 연체자의 비율(약 96:4)를 유지하도록 train : validation : test 데이터를 8:1:1로 층화추출을 수행하였다. 이후 train data로 학습하여 validation data에 test를 해보며 최적의 parameter를 찾았으며 test data로 최종적인 modeling 평가를 진행하였다.

02 Modeling

Baseline Model – SMOTE DBSCAN method + Gradient Boosting



Train data에서 SMOTE algorithm의 DBSCAN method로 upsampling하여 미연체자와 연체자의 비율을 약 1:1로 만들었다.이후 앙상블(Ensemble)기법인 gradient boosting을 통하여 예측

(SMOTE의 DBSCAN method은 KNN 대신에 DBSCAN clustering method으로 알고리즘을 수행)

→ Validation / Test F-measure : 0.43~0.49

다음 장에서 설명

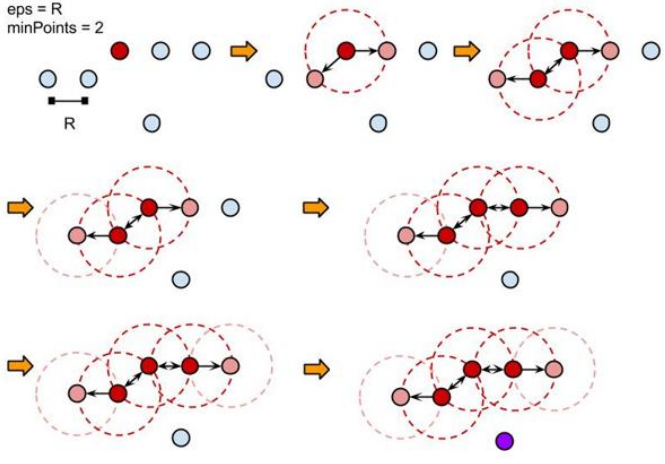
02 Modeling

※ DBSCAN clustering algorithm

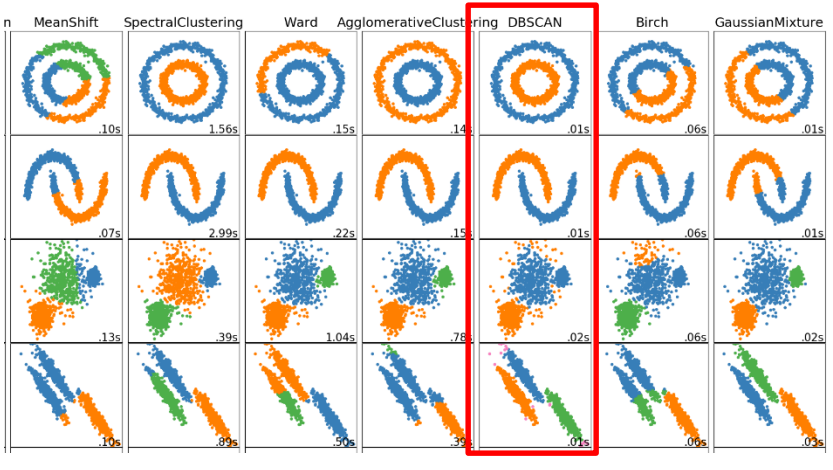
- Density-based Spatial Clustering of Applications with Noise의 약자
- 2014년 KDD 학회에서 상을 받은 알고리즘으로, density-based clustering 중 가장 성능이 우수함
- DBSCAN의 특징은 eps-neighbors와 MinPts(Minpoints)를 사용하여 군집을 구성

  Eps-neighbors: 한 데이터를 중심으로 epsilon(ϵ) 거리 이내의 데이터들을 한 군집으로 구성

  MinPts: 한 군집은 MinPts 보다 많거나 같은 수의 데이터로 구성됨 (MinPts는 보통 변수의 개수+1로 설정)
  만약 MinPts 보다 적은 수의 데이터가 eps-neighbors를 형성하면 노이즈(noise)로 취급함



DBSCAN 알고리즘 순서도



DBSCAN 알고리즘 성능비교평가

But, 문제가 있다!

1. Sampling에 따라서 validation과 Test의 F-measure perfomence의 분산이 매우 크다.
2. Y값을 0과 1로 표현해야 하는데, gradient boosting model은 Y값을 확률로 주어진다. 보통 0.5를 기준으로 threshold를 정하는데, data sampling에 따라서 최적의 thresehold값이 달라진다.

02 Modeling

Model strategy

- Model의 F-measure performence나 threshold가 sampling에 따라서 크게 다르다는 것은 model이 불안정하다는 것을 뜻하고, 이에 general한 model을 만들고자 크게 3가지 전략을 사용

1) 첫번째 전략

– 잘 맞추지 못하는 데이터를 greedy하게 지우는 전략

2) 두번째 전략

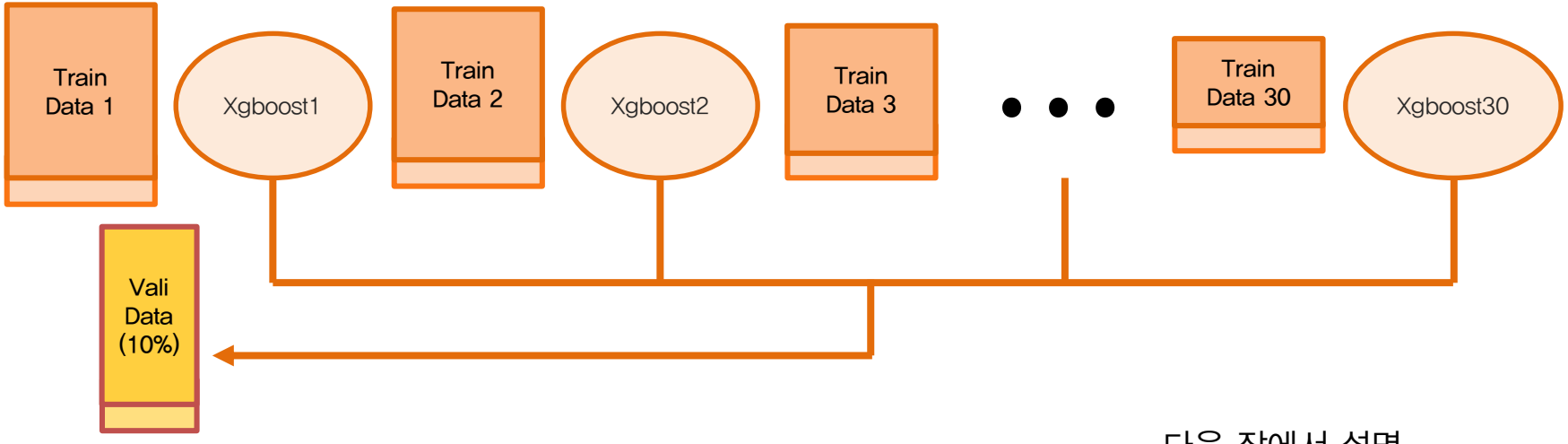
– 잘 맞추는 데이터를 greedy하게 늘리는 전략

3) 세번째 전략

– 첫번째 전략과 두번째 전략을 동시에 사용

02 Modeling

Model strategy 1 – Greedy Downsampling + Gradient Boosting



다음 장에서 설명

Step 1) i번째 data를 가지고 i번째 gradient boosting model을 만든다.

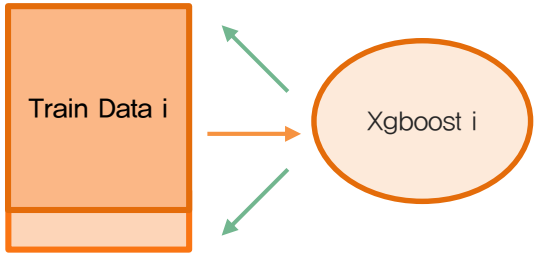
Step 2) gradient boosting model의 결과를 기반으로 greedy downsampling을 하여 i+1 번째 dataset을 만든다.

Step 3) Step1 2를 30회를 반복하여 총 30개의 gradient boosting model을 만들어 이를 bagging 모형과 같이 30개 model의 투표를 통해 validation data의 Y값을 예측한다.

02 Modeling

Model strategy 1(step2) – DBSCAN을 기반으로 한 Greedy Downsampling

1)



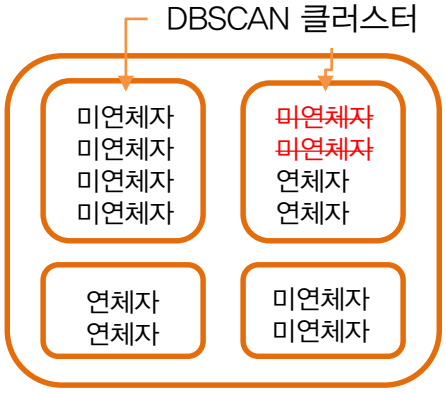
2)

	0	1
0	79520	1054
1	2032	2592

0– 미연체자

1– 연체자

3)



Step1) i번째 dataset을 가지고 gradient boosting으로 modeling을 한다. 이렇게 만들어진 model로 다시 train data에 대한 Y값을 예측한다.

Step 2) 예측한 train Data의 Y값 중 예측이 틀린 data들만 가지고 DBSCAN clustering을 진행한다.

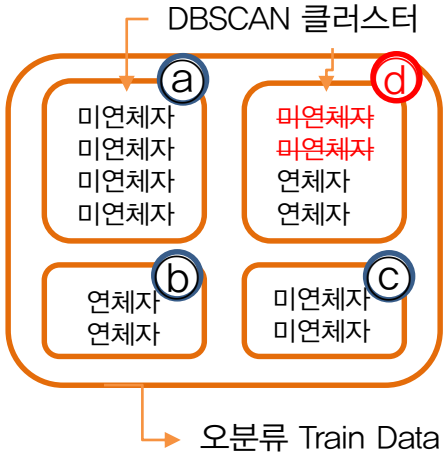
Step 3) 이 때, 미연체자와 연체자가 섞인 cluster가 존재하면 해당 cluster의 미연체자 data들만 제거하여 i+1번째 dataset을 만든다.

(즉, model을 만드는 데에 있어 decision boundary를 과적합 시킨다고 판단되는 data들을 제거하여 generalize model을 만들려는 과정이라고 할 수 있음)

02 Modeling

Model strategy 1(step2) – DBSCAN을 기반으로 한 Greedy Downsampling

3)



a b c 클러스터 : 오분류 되어있으나, 같은 클러스터안에 있으므로 재 학습하였을 때, 정분류 될 가능성이 높음

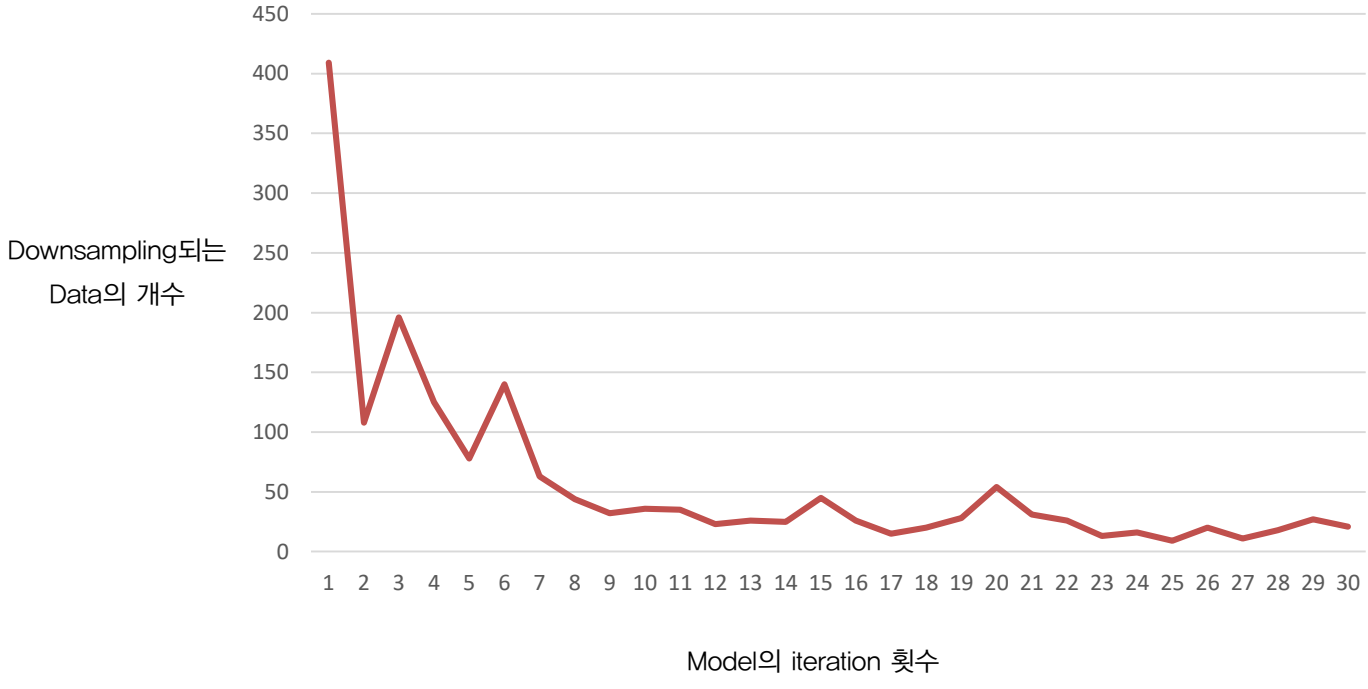
d 클러스터 : 오분류 되어있고, 소수 class인 연체자 데이터의 근처에 있으므로, 학습을 방해하는 데이터일 가능성이 높음 –〉 삭제

Step 3) 이 때, 미연체자와 연체자가 섞인 cluster가 존재하면 해당 cluster의 미연체자 data들만 제거하여 i+1번째 dataset을 만든다.

(즉, model을 만드는 데에 있어 decision boundary를 과적합 시킨다고 판단되는 data들을 제거하여 generalize model을 만들려는 과정이라고 할 수 있음)

02 Modeling

Model strategy 1 result



- 학습이 진행됨에 따라 downsampling되는 데이터의 수는 적어진다.
- 학습에 방해가 되는 데이터를 제거 함으로써 더 좋은 decision boundary생성을 가능토록함

02 Modeling

Model strategy 1 result

Validation threshold	Validation F-measure	Test F-threshold	Test F-measure
0.24	0.490196	0.195	0.49919
0.21	0.513912	0.245	0.489097
0.255	0.496392	0.215	0.51462
0.23	0.508029	0.23	0.530846
0.2	0.522857	0.205	0.519481
0.205	0.481818	0.2	0.505495
0.17	0.516383	0.225	0.481752
0.145	0.509863	0.225	0.485021
0.195	0.498003	0.22	0.493827
0.22	0.53902	0.25	0.516644
0.225	0.522367	0.23	0.478134
0.205	0.513219	0.215	0.519658
0.265	0.507163	0.3	0.495726
0.22	0.487106	0.25	0.515152
0.22	0.5	0.195	0.495413
0.265	0.507132	0.2	0.527851
0.245	0.490694	0.175	0.509972
0.21	0.48913	0.25	0.484018
0.23	0.5248	0.265	0.504298
0.26	0.479461	0.3	0.48494
mean	**0.504877**	**mean**	**0.502557**

- Data split부터 Model strategy 1을 20회 반복하여 낸 F-measure와 threshold는 왼쪽과 같다.
- 데이터에 따라 0.48~0.52 (평균 0.503)
- 데이터에 따라 성능이 널 뛰는 문제 완화 (Baseline 0.43~0.49)
- 평균 F-measure가 Baseline Model보다 성능이 약 2~3% 높음

학습 전(학습데이터의 수)

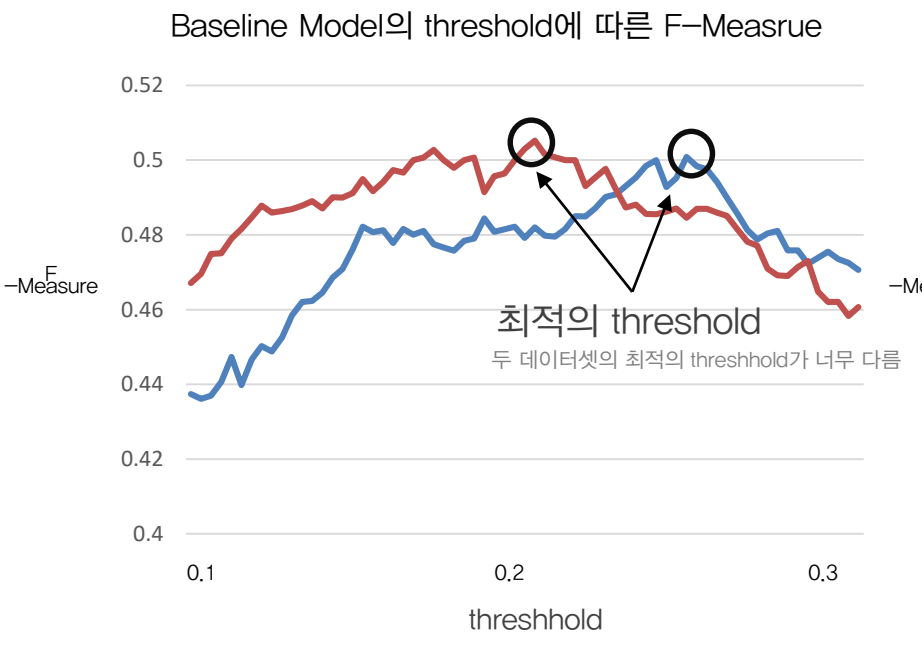
target	0	1
	76743	3443

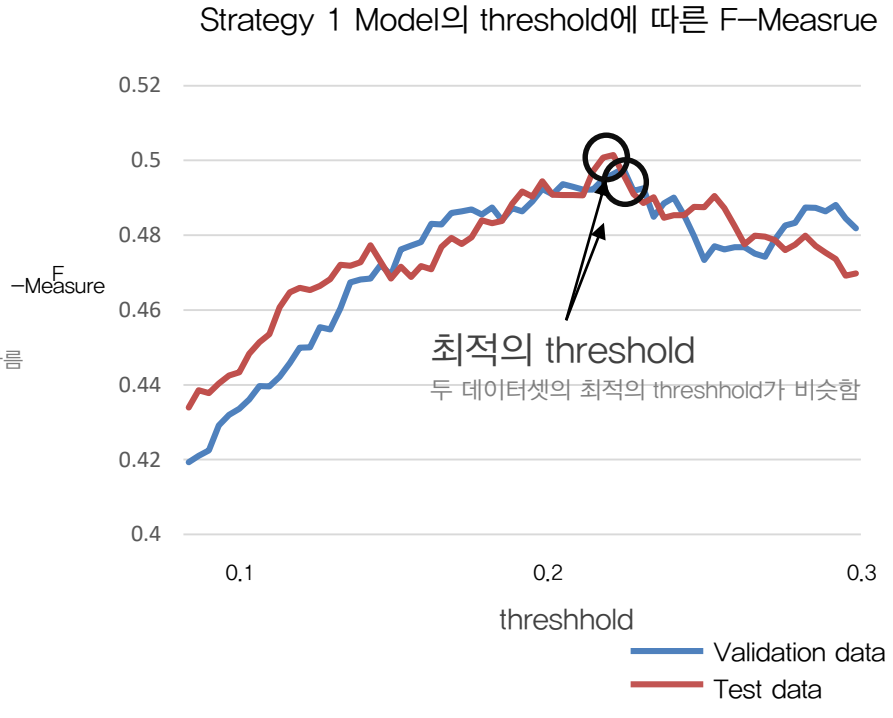
학습 후 — 약 7,000개의 데이터 감소

target	0	1
	66954	3443

02 Modeling

Model strategy 1 result





- 데이터에 따라 최적의 threshhold가 달라지는 달라 지는 문제 완화

02 Modeling

Model strategy

- Model의 F-measure performence나 threshold가 sampling에 따라서 크게 다르다는 것은 model이 불안정하다는 것을 뜻하고, 이에 generalize한 model을 만들고자 크게 3가지 전략을 사용

1) 첫번째 전략

– 잘 맞추지 못하는 데이터를 greedy하게 지우는 전략

2) 두번째 전략

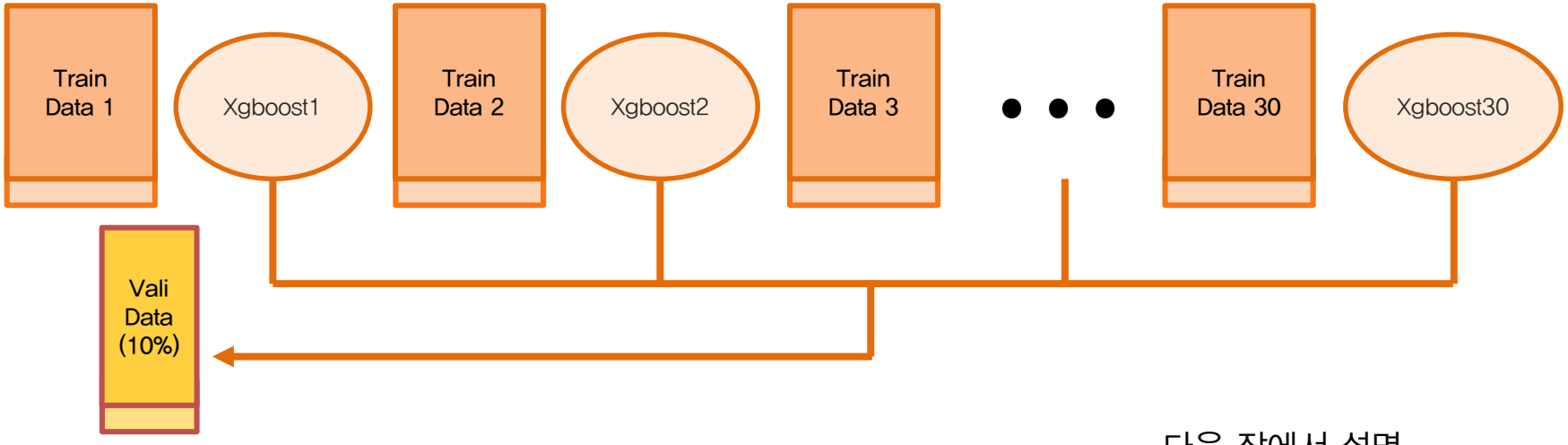
– 잘 맞추는 데이터를 greedy하게 늘리는 전략

3) 세번째 전략

– 첫번째 전략과 두번째 전략을 동시에 사용

02 Modeling

Model strategy 2 – Greedy Upsampling + Gradient Boosting



다음 장에서 설명

Step 1) i번째 data를 가지고 i번째 gradient boosting model을 만든다.

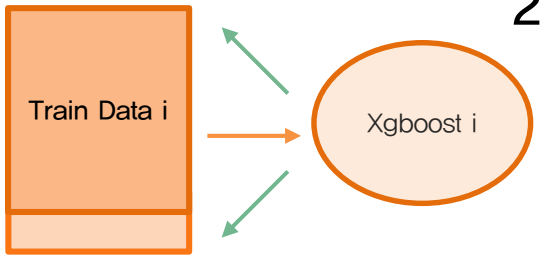
Step 2) gradient boosting model의 결과를 기반으로 greedy upsampling을 하여 i+1번째 dataset을 만든다.

Step 3) Step1 2를 30회를 반복하여 총 30개의 gradient boosting model을 만들어 이를 bagging 모형과 같이 30개 model의 투표를 통해 validation data의 Y값을 예측한다.

02 Modeling

Model strategy 2(step2) – SMOTE의 DBSCAN method 기반 Upsampling 방법

1)



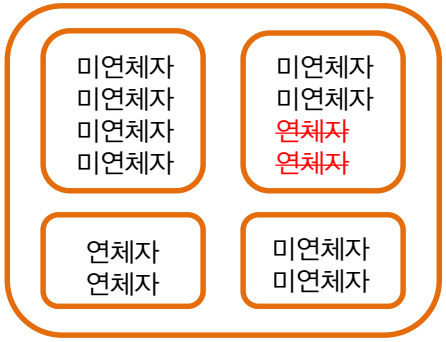
2, 4)

	0	1
0	79520	1054
1	2032	2592

0– 미연체자

1– 연체자

3)



Step1) i번째 Dataset을 가지고 i번째 gradient boosting을 진행하여 나온 model로 다시 train Data에 대한 Y값을 예측한다.

Step 2) Train Data의 Y값 예측이 틀린 Data들만 가지고 DBSCAN clustering을 진행한다.

Step 3) 이 때, 미연체자와 연체자가 섞인 cluster가 존재하면 downsampling 방법과 다르게 해당 cluster의 연체자 data를 제거한다.

Step 4) Step 2에서 잘 예측한 연체자들의 data를 SMOTE algorithm의 DBSCAN method를 이용해서 upsampling하여 i+1번째 dataset을 만든다.

(model을 만드는 데에 있어 decision boundary를 robust하게 만들어 generalize model을 만들려는 과정)

02 Modeling

Model strategy 2 result

Validation threshold	Validation F-measure	Test threshold	Test F-measure
0.23	0.487374	0.195	0.495413
0.26	0.515913	0.24	0.489362
0.26	0.487261	0.255	0.521246
0.25	0.508221	0.205	0.52819
0.21	0.517028	0.26	0.516667
0.185	0.47619	0.3	0.512156
0.165	0.519757	0.22	0.50514
0.2	0.49553	0.185	0.5
0.22	0.499291	0.27	0.504559
0.23	0.543575	0.21	0.527228
0.21	0.5171	0.25	0.484245
0.165	0.527301	0.23	0.503864
0.26	0.498734	0.26	0.497207
0.27	0.486068	0.225	0.516691
0.18	0.491803	0.21	0.490798
0.225	0.504043	0.17	0.533865
0.19	0.496329	0.24	0.505952
0.14	0.494935	0.21	0.473595
0.3	0.522003	0.3	0.502463
0.205	0.478983	0.21	0.48103
mean	**0.503372**	**mean**	**0.504483**

- Data split부터 Model strategy 2를 20회 반복하여 낸 F-measure와 threshold는 왼쪽과 같다.
- 데이터에 따라 0.48~0.52 (평균 0.503)
- 평균 F-measure가 Baseline Model 보다 성능이 약 2~3% 높음

학습 전(학습데이터의 수)

target	0	1
	76743	3443

학습 후

약 21,000개의 데이터 증가

target	0	1
	76743	25046

02 Modeling

Model strategy

- Model의 F-measure performence나 threshold가 sampling에 따라서 크게 다르다는 것은 model이 불안정하다는 것을 뜻하고, 이에 generalize한 model을 만들고자 크게 3가지 전략을 사용

1) 첫번째 전략

– 잘 맞추지 못하는 데이터를 greedy하게 지우는 전략

2) 두번째 전략

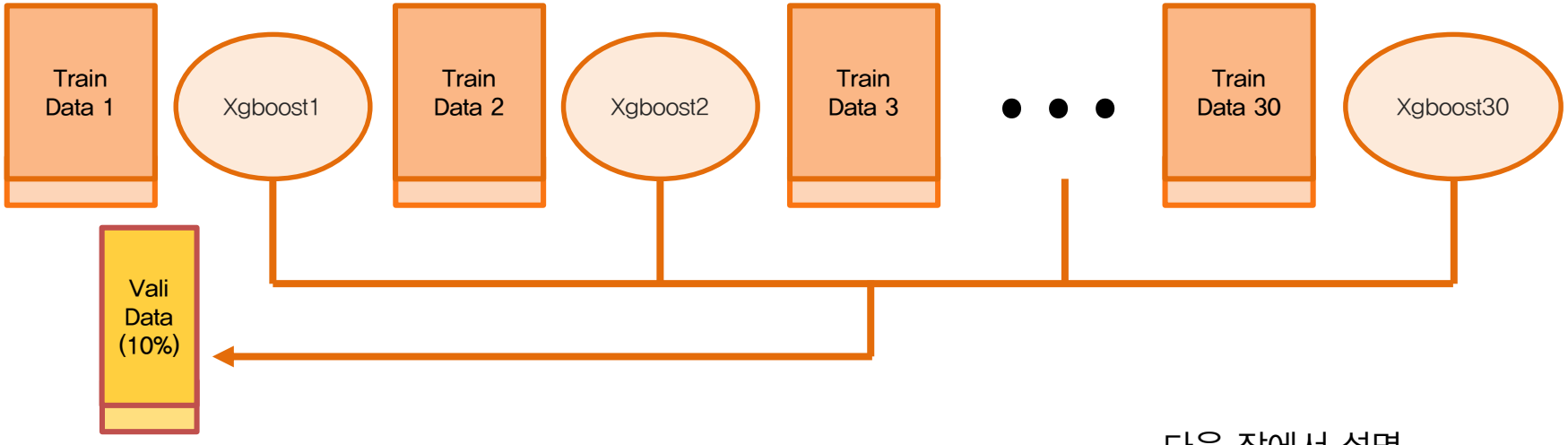
– 잘 맞추는 데이터를 greedy하게 늘리는 전략

3) 세번째 전략

– 첫번째 전략과 두번째 전략을 동시에 사용

02 Modeling

Model strategy 3 – Greedy Up&down sampling + Gradient Boosting



다음 장에서 설명

Step 1) i번째 data를 가지고 i번째 gradient boosting model을 만든다.

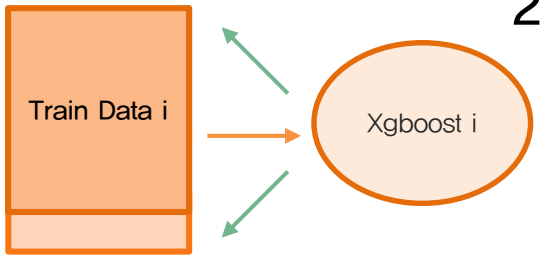
Step 2) gradient boosting model의 결과를 기반으로 greedy up&down sampling을 하여 i+1번째 dataset을 만든다.

Step 3) Step1 2를 30회를 반복하여 총 30개의 gradient boosting model을 만들어 이를 bagging 모형과 같이 30개 model의 투표를 통해 validation data의 Y값을 예측한다.

02 Modeling

Model strategy 3(step2) – SMOTE의 DBSCAN 방식 기반 Up&down sampling 방법

1)



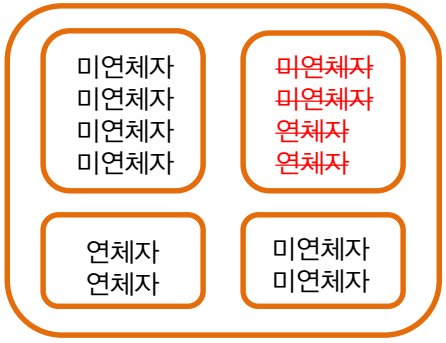
2, 4)

	0	1
0	79520	1054
1	2032	2592

0– 미연체자

1– 연체자

3)



Step1) i번째 dataset을 가지고 i번째 gradient boosting을 진행하여 나온 model로 다시 train data에 대한 Y값을 예측한다.

Step 2) Train data의 Y값 중 예측이 틀린 data들만 가지고 DBSCAN clustering을 진행한다.

Step 3) 미연체자와 연체자가 섞인 cluster가 존재하면 앞선 두 방법과 다르게 해당 Cluster data를 모두 제거한다.

Step 4) Step 2에서 잘 예측한 연체자들의 data를 SMOTE의 DBSCAN method를 이용해서 upsampling하여 i+1번째 dataset을 만든다.

(앞선 두 방법들의 효과를 동시에 보고자 함)

02 Modeling

Model strategy 3 result

Validation threshold	Validation F-measure	Test F-threshold	Test F-measure
0.23	0.493606	0.195	0.514505
0.26	0.514443	0.24	0.487805
0.26	0.492041	0.255	0.520674
0.25	0.51719	0.235	0.527407
0.21	0.522796	0.245	0.508475
0.195	0.489796	0.255	0.518987
0.165	0.532189	0.22	0.502128
0.2	0.506122	0.185	0.494605
0.22	0.499363	0.27	0.504644
0.23	0.533537	0.21	0.53973
0.21	0.527778	0.25	0.497966
0.165	0.517241	0.23	0.513475
0.26	0.513981	0.26	0.490399
0.27	0.486608	0.225	0.523605
0.195	0.5	0.21	0.491803
0.225	0.513995	0.19	0.534884
0.19	0.491935	0.24	0.502046
0.18	0.483461	0.21	0.483063
0.3	0.519685	0.3	0.52
0.205	0.506071	0.21	0.491642
mean	**0.508092**	**mean**	**0.508392**

- Data split부터 Model strategy 3을 20회 반복하여 낸 F-measure와 threshold는 왼쪽과 같다.
- 데이터에 따라 0.48~0.52 (평균 0.508)
- 평균 F-measure가 Baseline Model보다 성능이 약 3~4% 높음

학습 전(학습데이터의 수)

target	0	1
	76743	3443

학습 후

약 22,000개의 데이터 증가

target	0	1
	66316	26083

약 10,000개의 데이터 감소

03 Model assessment

F-measure를 통한 모델 비교

	F-measure
Baseline Model	0.45
Proposed Model1	0.503
Proposed Model2	0.503
Proposed Model3	0.508

– Proposed method 간에 큰 성능 차이는 없으나, Validation의 평균 F–Measure가 가장 높은 Proposed Model3 (전략3) 선택

예상 Test F–measure : 0.48~0.53

(Test 데이터수가 Train데이터수의 약 0.2% 수준으로 데이터에 따라 성능이 다르게 측정되리라 예상)

04 Summary

① Gradient Boosting 방법을 기반으로 한 파생변수 추출

② Data를 가지고 Gradient Boosting Modeling 진행

③ DBSCAN을 활용한 Greedy Data Up&down sampling
(Greedy 하게 sampling하는것은 모형을 안정화시키는데 효과가 있다)

④ Up&down sampling된 Data를 가지고 Gradient Boosting Modeling (ensemble)

⑤ 순차적으로 만들어진 30개의 Gradient Boosting Model을 Bagging기법을 통해 예측하여 모형의 예측력 및 정확도 향상 (안정성 확보)

05 추가실험

– Synthetic Data (2차원 class imbalanced 가상 데이터)

Positive ratio : 20%
8:2 분할 후 10회 반복 하여 모델 비교 실험

	단일 Model (Decision Tree)	smote	dbscan	Proposed Method 1	Proposed Method 2	Proposed Method 3
1	0.6452	0.652	0.6604	**0.6832**	0.6452	**0.6832**
2	0.6957	0.7012	0.6992	**0.7024**	0.6957	**0.7024**
3	0.6773	0.6821	0.6993	**0.7005**	0.6833	0.6965
4	0.6683	0.6542	0.6926	**0.7019**	0.6683	0.7012
5	0.6512	0.6358	0.682	0.6988	0.6588	**0.7003**
6	0.6715	0.6914	0.6956	0.714	0.7072	**0.7145**
7	0.6517	0.6725	0.6701	0.6807	0.6734	**0.683**
8	0.7244	0.6995	0.6953	**0.7314**	0.7214	0.7268
9	0.6783	0.7001	0.6838	**0.7064**	0.6995	0.7057
10	0.6452	0.6521	0.705	**0.7288**	0.6941	0.7163
평균	0.67088	0.67409	0.68833	**0.70474**	0.68469	0.70299

05 추가실험

– Synthetic Data (2차원 class imbalanced 가상 데이터)

Positive ratio : 10%
8:2 분할 후 10회 반복 하여 모델 비교 실험

	단일 Model (Decision Tree)	smote	dbscan	Proposed Method 1	Proposed Method 2	Proposed Method 3
1	0.4676	0.4675	0.5572	0.5501	0.5596	**0.5803**
2	0.4396	0.4421	0.5273	**0.5629**	0.5419	0.5491
3	0.3902	0.4525	0.5169	**0.5532**	0.525	0.5381
4	0.4604	0.5012	**0.5467**	0.5241	0.5155	0.5442
5	0.4	0.5123	0.5175	0.5348	0.519	**0.5469**
6	0.4099	0.5532	0.5405	**0.5691**	0.5612	0.5614
7	0.4183	0.4536	0.5524	0.5542	0.5547	**0.558**
8	0.4463	0.4958	0.534	0.539	0.5315	**0.5658**
9	0.4172	**0.5421**	0.5366	0.5264	0.52	0.5293
10	0.4568	0.5562	**0.5679**	0.5272	0.5534	0.5505
평균	0.43063	0.49765	0.5397	0.5441	0.53818	**0.55236**

05 추가실험

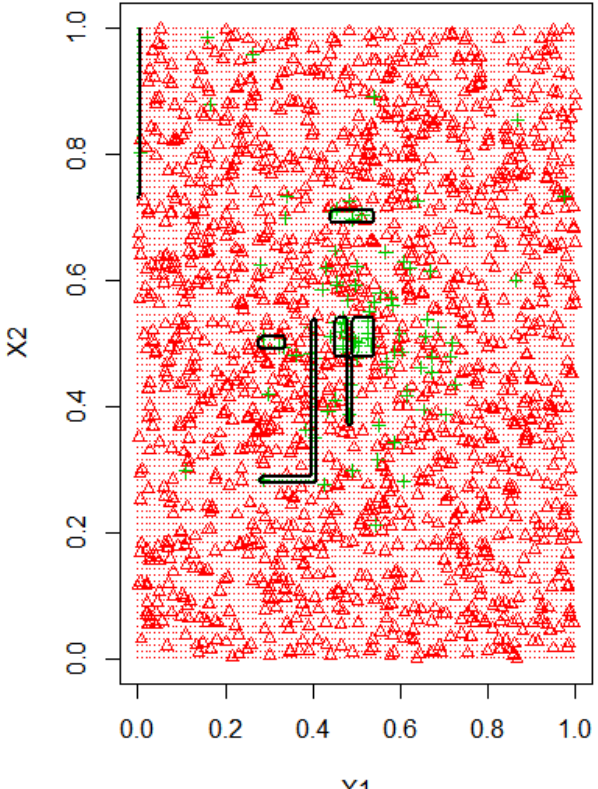
– Synthetic Data (2차원 class imbalanced 가상 데이터)

Positive ratio : 5%
8:2 분할 후 10회 반복 하여 모델 비교 실험

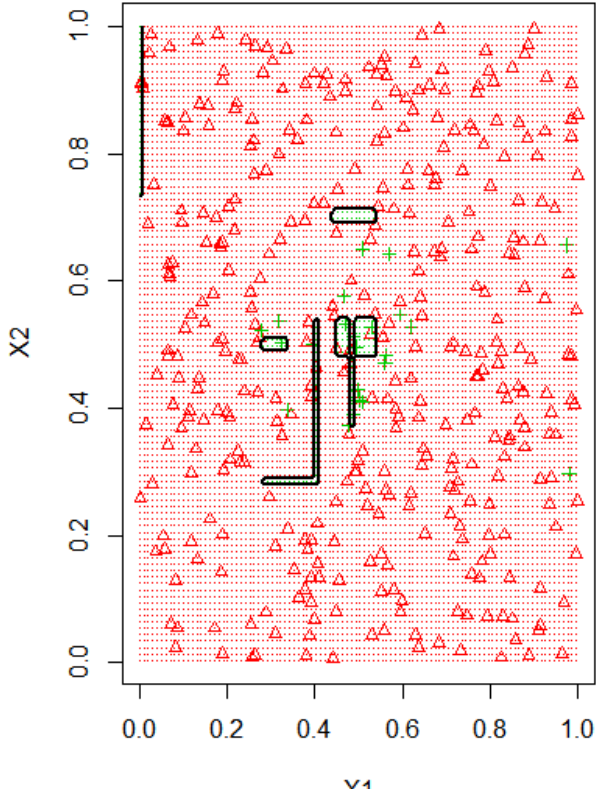
	단일 Model (Decision Tree)	smote	dbscan	Proposed Method 1	Proposed Method 2	Proposed Method 3
1	0.2154	0.3025	0.328	0.3368	0.3547	**0.3566**
2	0.2014	0.3245	0.4358	0.3448	0.4288	**0.4301**
3	0.2568	0.4215	0.4687	0.4027	0.4754	**0.4792**
4	0.2015	0.3521	0.3775	0.3435	**0.3894**	0.3692
5	0.3681	0.3402	0.415	0.3824	**0.4184**	0.4182
6	0.1958	0.2987	0.416	0.2973	0.4218	**0.4335**
7	0.2857	0.3245	0.3901	0.3524	0.4001	**0.4039**
8	0.1746	0.3678	0.4081	0.3555	**0.4302**	0.4293
9	0.2567	0.4001	0.3978	0.3668	0.4047	**0.435**
10	0.2353	0.3987	0.4484	0.2881	**0.4487**	0.4486
평균	0.23913	0.35306	0.40854	0.34703	0.41722	**0.42036**

05 추가실험

- Decision boundary (Synthetic Data, positive ratio 5%)
  Base model(Decision Tree)로 기존데이터에 학습한 후 Decision boundary 시각화



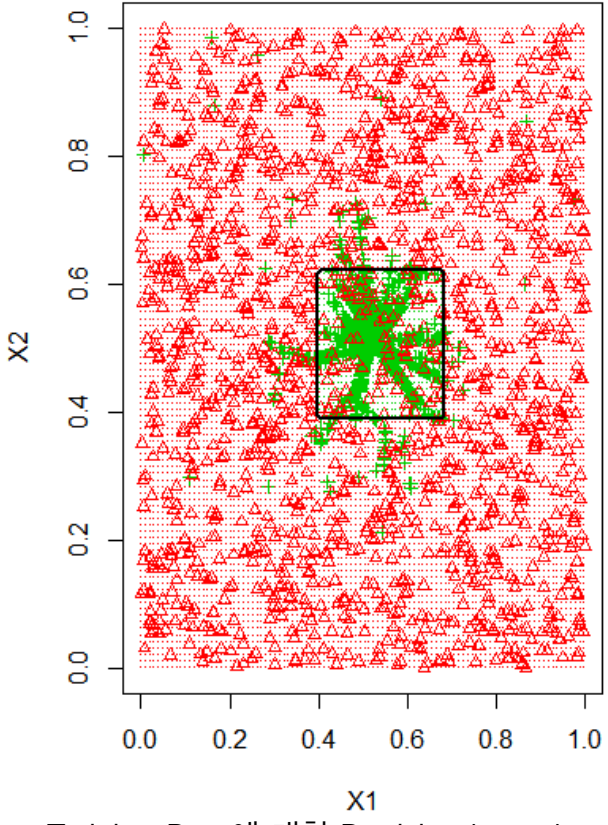
Training Data에 대한 Decision boundary



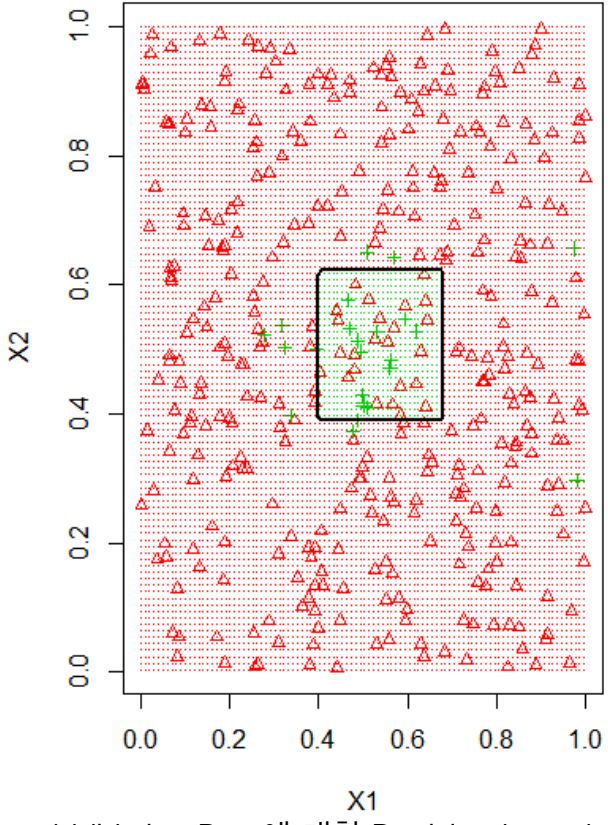
Validation Data에 대한 Decision boundary (**Fm 0.21**)

05 추가실험

- Decision boundary (Synthetic Data, positive ratio 5%)
  DBSCAN SMOTE를 하여 학습한 후 Decision boundary 시각화



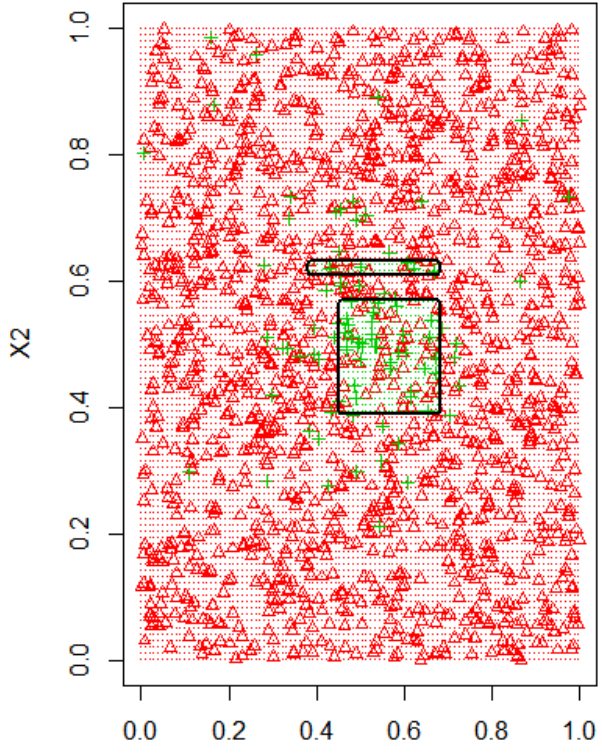
Training Data에 대한 Decision boundary



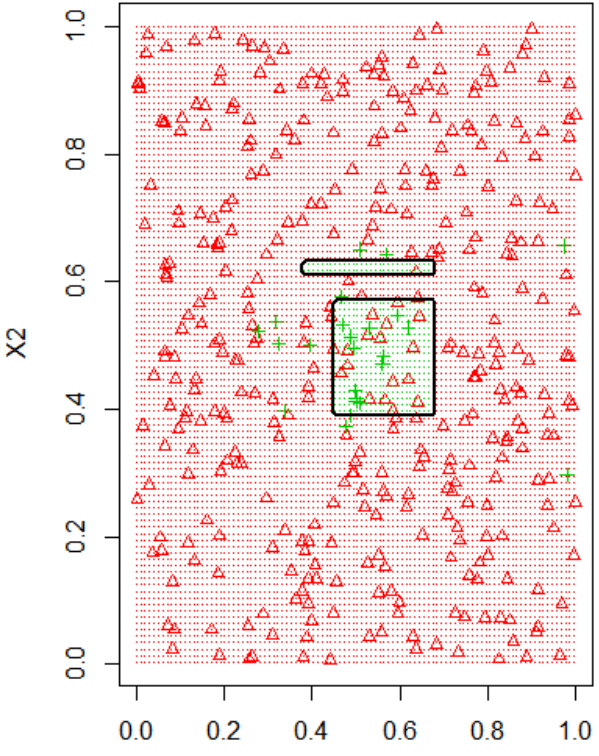
Validation Data에 대한 Decision boundary (**Fm 0.47**)

05 추가실험

- Decision boundary (Synthetic Data, positive ratio 5%)
  Proposed method1 (못 맞추는 0의 데이터 제거) 를 통하여 학습한 후 Decision boundary 시각화



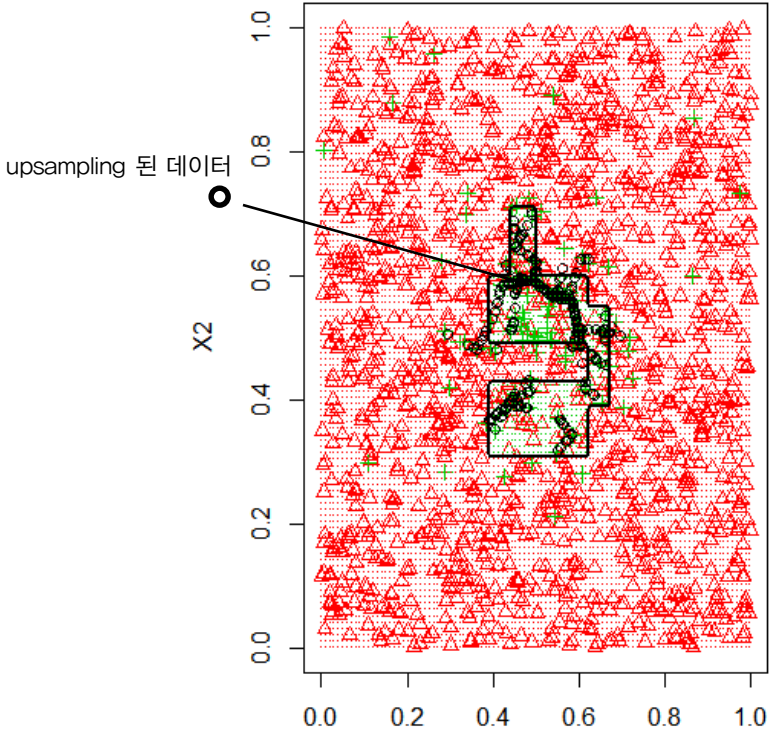
Training Data에 대한 Decision boundary



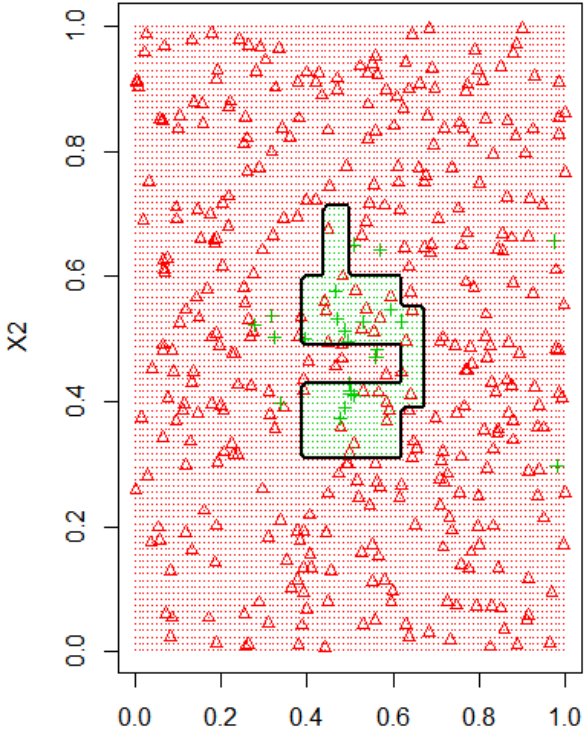
Validation Data에 대한 Decision boundary (Fm 0.48)

05 추가실험

- Decision boundary (Synthetic Data, positive ratio 5%)
  Proposed method2 (잘 맞추는 1의 데이터 upsampling) 를 통하여 학습한 후 Decision boundary 시각화



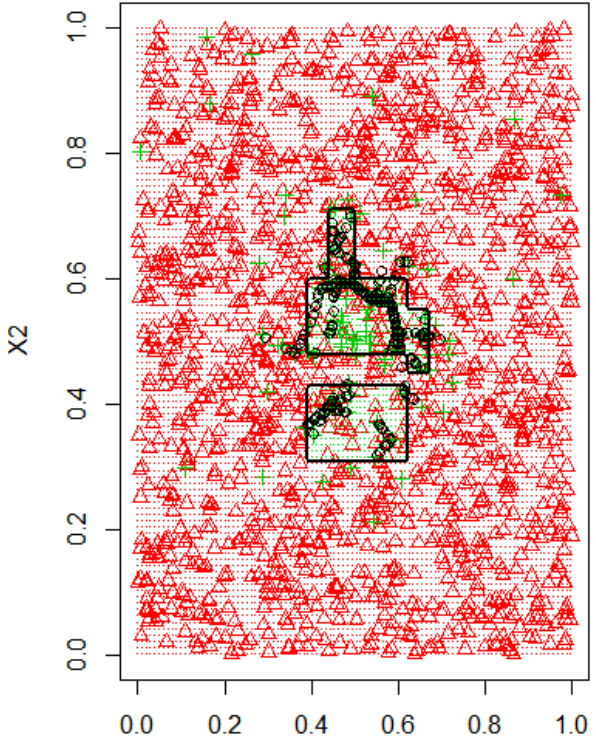
Training Data에 대한 Decision boundary



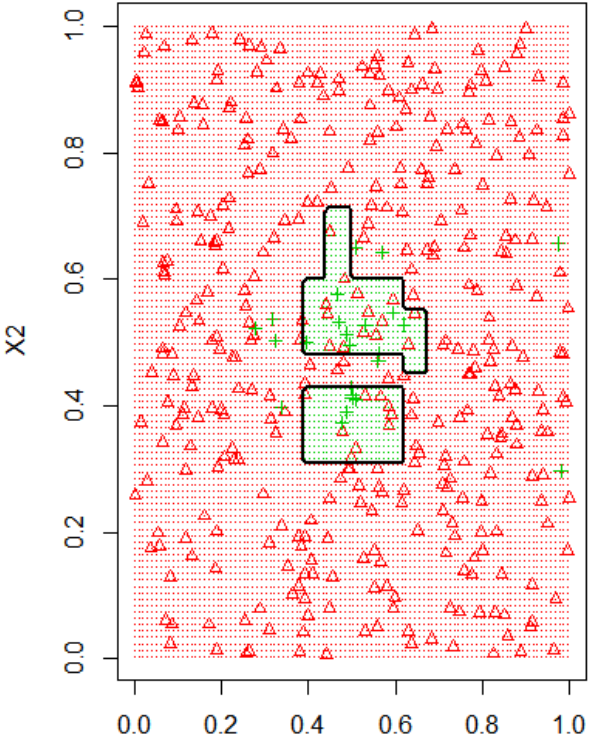
Validation Data에 대한 Decision boundary (Fm 0.48)

05 추가실험

- Decision boundary (Synthetic Data, positive ratio 5%)
  Proposed method3 (up&down sampling) 를 통하여 학습한 후 Decision boundary 시각화



Training Data에 대한 Decision boundary



Validation Data에 대한 Decision boundary (Fm 0.49)

05 추가실험

– Kaggle Data

Data Name : Santander Customer Satisfaction (Positive ratio : 3.95%)
8:2 분할 후 10회 반복 하여 모델 비교 실험

	단일 model(Xgboost)		SMOTE		DBSCAN SMOTE		**Proposed Method3**	
횟수	F-measure	AUC	F-measure	AUC	F-measure	AUC	F-measure	AUC
1	0.249697	**0.812339**	0.248293	0.808472	0.244996	0.805155	**0.271293**	0.806993
2	0.263126	0.807249	0.271028	0.81113	0.247379	0.802865	**0.280098**	**0.817196**
3	0.256019	0.818402	0.249272	0.815489	0.257348	0.821501	**0.266033**	**0.821964**
4	0.241785	0.806577	0.248258	0.805867	0.242507	0.799921	**0.262177**	**0.807775**
5	0.267835	0.814963	**0.29644**	**0.826919**	0.277728	0.815067	0.287807	0.823878
6	0.262061	0.788997	0.240469	0.795573	0.25293	0.794348	**0.279967**	**0.813024**
7	0.257402	0.810502	0.260819	0.812138	0.242678	0.81025	**0.270997**	**0.816147**
8	0.258767	0.803529	0.246675	0.79453	0.248184	0.798238	**0.284302**	**0.803782**
9	0.250147	0.796047	0.240272	0.801707	0.256342	0.79124	**0.262102**	**0.804724**
10	0.259373	0.804144	0.238263	0.79971	0.244076	0.798422	**0.261364**	**0.801463**
평균	0.256621	0.806275	0.253979	0.807153	0.251417	0.803701	**0.272614**	**0.811694**

05 추가실험

– Kaggle Data

Data Name : Safe Driver Prediction (Positive ratio : 3.64%)
8:2 분할 후 10회 반복 하여 모델 비교 실험

	단일 model(Xgboost)		SMOTE		DBSCAN SMOTE		**Proposed Method3**	
횟수	F-measure	AUC	F-measure	AUC	F-measure	AUC	F-measure	AUC
1	0.07641	0.597783	0.094466	0.610927	0.084093	0.591002	**0.109317**	**0.614703**
2	0.069522	0.60849	0.094875	**0.609185**	0.076113	0.594371	**0.11039**	0.60898
3	0.066926	0.610001	0.106985	**0.617217**	0.071436	0.594318	**0.110085**	0.612211
4	0.065966	0.60506	0.105083	0.615588	0.072045	0.595235	**0.108748**	**0.611086**
5	0.068247	0.601185	0.09132	0.599499	0.086344	0.60681	**0.107949**	**0.611122**
6	0.066406	0.603209	0.093557	0.605391	0.071386	**0.610062**	**0.110983**	0.609179
7	0.069403	0.59499	0.102528	0.600931	0.086526	0.597814	**0.108984**	**0.610498**
8	0.068003	0.603126	0.101996	0.605259	0.085164	**0.608334**	**0.104838**	0.607794
9	0.062946	0.608238	0.098235	0.605351	0.074123	0.591827	**0.107102**	**0.609235**
10	0.070912	0.595413	**0.108381**	0.597995	0.086507	0.599272	0.103	**0.61168**
평균	0.068474	0.60275	0.099743	0.607534	0.079374	0.598904	**0.10814**	**0.610649**

05 추가실험

- 2차원 가상 데이터를 통해 Decision boundary가 좋아지는 것을 확인
- 대회 데이터 뿐 아니라, class imbalanced data에 모두 적용 가능 한 방법론

Thank You

06 Codebook

파생변수 생성 코드

```
psall$add_var6<-psall[,8]/psall[,6]
psall$add_var7<-psall[,9]/psall[,6]
psall$add_var8<-psall[,10]/psall[,6]
psall$add_var9<-psall[,7]/psall[,6]
psall$add_var10<-psall$cb_guif_amt/psall$cust_job_incm
psall$add_var11<-psall$cb_guif_cnt/psall$cust_job_incm
psall$add_var12<-psall[,8]/psall$cust_job_incm
psall$add_var13<-psall[,9]/psall$cust_job_incm
psall$add_var14<-psall[,10]/psall$cust_job_incm
psall$add_var15<-psall[,11]/psall$cust_job_incm
psall$add_var16<-psall$cb_guif_amt/psall$hshd_infr_incm
psall$add_var17<-psall$cb_guif_cnt/psall$hshd_infr_incm
psall$add_var18<-psall[,8]/psall$hshd_infr_incm
psall$add_var19<-psall[,9]/psall$hshd_infr_incm
psall$add_var20<-psall[,10]/psall$hshd_infr_incm
psall$add_var21<-psall[,11]/psall$hshd_infr_incm
psall$add_var22<-psall$cb_guif_amt/psall$cust_fmly_num
psall$add_var23<-psall$cb_guif_cnt/psall$cust_fmly_num
psall$add_var24<-psall[,8]/psall$cust_fmly_num
psall$add_var25<-psall[,9]/psall$cust_fmly_num
psall$add_var26<-psall[,10]/psall$cust_fmly_num
psall$add_var27<-psall[,11]/psall$cust_fmly_num
psall$add_var28<-psall$strt_crdt_grad * psall$crln_30ovdu_rate
psall$add_var29<-psall$ltst_crdt_grad * psall$lt1y_clod_rate
psall$add_var30<-psall$strt_crdt_grad * psall$crln_30ovdu_rate
psall$add_var31<-psall$strt_crdt_grad * psall$lt1y_clod_rate
psall$add_var32<-psall$tot_prem / psall$cust_job_incm
psall$add_var33<-psall$fmly_tot_prem / psall$cust_job_incm
psall$add_var34<-psall$fmly_tot_prem / psall$hshd_infr_incm
psall$add_var35<-psall$strt_crdt_grad * psall$lt1y_ctlt_cnt
psall$add_var36<-psall$fmly_clam_cnt * psall$tlfe_unpd_cnt
psall$add_var37<-psall$cntt_lamt_cnt * psall$tlfe_unpd_cnt
psall$add_var38<- psall$sex1*psall$avg_call_time
psall$add_var39<- psall$sex2*psall$avg_call_time
psall$add_var40<-psall$mon_tlfe_amt/psall$fmly_tot_prem
psall$add_var41<-psall$mon_tlfe_amt/psall$tot_prem
psall$add_var42<-psall$mon_tlfe_amt/psall$tot_lnif_amt
psall$add_var43<-psall$mon_tlfe_amt/psall$tot_clif_amt
psall$add_var44<-psall$mon_tlfe_amt/psall$bnk_lnif_amt
psall$add_var45<-psall$mon_tlfe_amt/psall$cpt_lnif_amt
psall$add_var46<- psall$sex3*psall$avg_call_time
psall$add_var47<-psall$crmm_ovdu_amt * psall$tlfe_unpd_cnt
psall$add_var48<-psall$lt1y_mxod_amt * psall$tlfe_unpd_cnt
psall$add_var49<-psall$fycm_paid_amt/psall$tot_lnif_amt
psall$add_var50<-psall$fycm_paid_amt/psall$tot_clif_amt
psall$add_var51<-psall$fycm_paid_amt/psall$bnk_lnif_amt
psall$add_var52<-psall$fycm_paid_amt/psall$cpt_lnif_amt
psall$add_var100<- psall$avg_call_time*psall$strt_crdt_grad
psall$add_var101<- psall$fmly_clam_cnt/psall$strt_crdt_grad
psall$add_var102<-psall$fmly_clam_cnt/psall$ltst_crdt_grad
psall$add_var103<-psall$strt_crdt_grad * psall$lt1y_ctlt_cnt
psall$add_var104<-psall$tot_prem / psall$hshd_infr_incm
psall$add_var105<-psall$fycm_paid_amt /  psall$tot_prem
psall$add_var106<-psall$fycm_paid_amt /  psall$fmly_tot_prem
psall$add_var107<-psall$fmly_clam_cnt /  psall$cust_job_incm
psall$add_var108<-psall$fmly_clam_cnt /  psall$hshd_infr_incm
psall$add_var109<-psall$fycm_paid_amt /  psall$cust_job_incm
psall$add_var110<-psall$fycm_paid_amt /  psall$hshd_infr_incm
psall$add_var111<-psall$ltst_crdt_grad * psall$autr_fail_mcnt
psall$add_var112<-psall$ltst_crdt_grad * psall$lt1y_ctlt_cnt
psall$add_var113<-psall$strt_crdt_grad * psall$cntt_lamt_cnt
psall$add_var114<-psall$fmly_tot_prem / psall$strt_crdt_grad
psall$add_var115<-psall$fmly_tot_prem / psall$ltst_crdt_grad
psall$add_var116<-psall$ltst_crdt_grad * psall$tot_prem
psall$add_var117<-psall$ltst_crdt_grad * psall$tot_prem
psall$add_var118<-psall$max_mon_prem / psall$cust_job_incm
psall$add_var119<-psall$max_mon_prem / psall$hshd_infr_incm
psall$add_var120<-psall$ltst_crdt_grad * psall$max_mon_prem
psall$add_var121<-psall$strt_crdt_grad * psall$max_mon_prem
psall$add_var122<-psall$ltst_crdt_grad * psall$gdins_mon_prem
psall$add_var123<-psall$strt_crdt_grad * psall$gdins_mon_prem
psall$add_var124<-psall$stln_remn_amt / psall$cust_job_incm
psall$add_var125<-psall$lt1y_stln_amt / psall$cust_job_incm
psall$add_var126<-psall$tot_repy_amt/psall$tot_crln_amt
psall$add_var127<-psall$add_var28/psall$cust_job_incm
psall$add_var128<-psall$add_var28/psall$hshd_infr_incm
psall$add_var129<-psall$cb_guif_amt/psall$mate_job_incm
psall$add_var130<-psall$cb_guif_cnt/psall$mate_job_incm
psall$add_var131<-psall[,8]/psall$mate_job_incm
psall$add_var132<-psall[,9]/psall$mate_job_incm
psall$add_var133<-psall[,10]/psall$mate_job_incm
psall$add_var134<-psall[,11]/psall$mate_job_incm
```